基于均衡数据放置策略的分布式网络存储编码缓存方案 *

陈　雪 [1]，胡玉平 [2]

(1. 广州工商学院 计算机科学与工程系，广州 510850; 2. 广东财经大学 信息学院，广州 510320)

摘　要：为了保证网络存储的负载平衡并避免在节点或磁盘故障的情况下造成不可恢复的损失，提出一种基于均衡数据放置策略的分布式网络存储编码缓存方案，针对大型高速缓存和小型缓存分别给出了不同的解决办法。首先，将 Maddah 方案扩展到多服务器系统，结合均衡数据放置策略，将每个文件作为一个单元存储在数据服务器中，从而解决大型高速缓存问题；然后，将干扰消除方案扩展到多服务器系统，利用干扰消除方案降低缓存的峰值速率，结合均衡数据放置策略，提出缓存分段的线性组合，从而解决小型缓存问题。最后，通过基于 Linux 的 NS2 仿真软件，分别在一个和两个奇偶校验服务器系统中进行仿真实验，仿真结果表明，提出的方案可以有效地降低峰值传输速率，相比其他两种较新的缓存方案，提出的方案获得了更好的性能。此外，采用分布式存储虽然限制了将来自不同服务器的内容组合成单个消息的能力，导致编码缓存方案性能损失，但可以充分利用分布式存储系统中存在的固有冗余，从而提高存储系统的性能。

关键词：均衡数据放置策略；分布式网络存储；编码缓存；文件条带化；奇偶校验服务器；干扰消除

中图分类号：TP393.07　　doi: 10.19734/j.issn.1001-3695.2018.11.0836

Distributed network storage and coding scheme based on balanced data placement strategy

Chen Xue[1], Hu Yuping[2]

(1. *Dept. of Computer Science & Engineering*, *Guangzhou College of Technology & Business*, *Guangzhou* 510850, *China*; 2. *School of Information Science*，*Guangdong University of Finance & Economics*, *Guangzhou* 510320, *China*)

Abstract: In order to ensure the load balance of network storage and to avoid unrecoverable loss in case of node or disk failures, this paper proposed a distributed storage and coding scheme based on balanced data placement strategy, which gave different solutions for large caches and small caches. Firstly, the Maddah scheme is extended to multi-server system, and each file is stored as a unit in the data server by combining balanced data placement strategy to solve the large-scale cache problem. Then, the interference cancellation scheme is extended to the multi-server system. The interference cancellation scheme is used to reduce the peak rate of the cache, and the linear combination of cache segments is proposed by combining balanced data placement strategy so as to solving the problem of small caching. Finally, simulation experiments are carried out in one and two parity check server systems respectively through Linux-based NS2 simulation software. The simulation results show that the proposed scheme can effectively reduce the peak transmission rate. It has achieved better performance comparing with the other two new caching schemes. In addition, although distributed storage limits the ability to combine content from different servers into a single message, resulting in performance loss of coding and caching schemes, it can make full use of the inherent redundancy in distributed storage systems, thereby improving the performance of storage systems.

Key words: balanced data placement strategy; distributed network storage; coded cache; file striping; parity check server; interference cancellation

0　引言

近年来，独立磁盘冗余阵列(redundant array of independent disks,RAID)[1,2]在编码缓存方面的应用越来越广泛，RAID 是一种基于纠删码的分布式存储技术，它将多个存储节点(磁盘、服务器等)组合成一个具有数据冗余的逻辑单元。其中两个最常见的是 RAID-4 和 RAID-6，分别由一个和两个专用奇偶校验节点组成的块级条带组成[3,4]。大多数大型系统使用某种形式的 RAID，将多个存储驱动器条带化，但将整个文件作为一个单元存储或复制到网络节点[5]。这样可以提高峰值速度，同时也简化了记录和重复数据删除，提高了安全性，并使网络更加灵活。文献[6]的研究关注一组具有本地存储器的用户如何通过一个共同的服务器从一台服务器有效地接收数据链接，它提出了一个缓存和交付方案，在信息理论最优的常数因子以及该最优值的上限和下限内提供最差情况下的性能。文献[7]对下界进行了细化，并设计了新的方案来考虑非统一的文件大小。文献[8]对高速缓存和传输段进行编码设计了一种新的算法，以获得比当缓存容量较小或用户数量大于文件数量时更好的性能。但是，这个方案着重于单个服务器系统。

在分布式存储方面，研究人员研究了单个用户如何能够有效地恢复分布在一组节点上的数据，并且研究了具有本地存储器的一组用户如何有效地从单个节点接收数据。为了保证网络存储的负载平衡并避免在节点或磁盘故障的情况下造

收稿日期：2018-11-30；**修回日期：**2019-01-21　　**基金项目：**广东省自然科学基金资助项目（2016A030313717）

作者简介：陈雪（1982-），女，湖北公安人，讲师，硕士，主要研究方向为图像处理、智能算法等（chen132898xu532@163.com）；胡玉平（1969-），男，湖南双峰人，教授，博士，主要研究方向为分布式存储、信息安全等.

成不可恢复的损失，提出一种基于均衡数据放置策略的分布式网络存储编码缓存方案，针对大型高速缓存和小型缓存分别给出了不同的解决办法。提出的方案旨在为多服务器多用户系统设计一个联合存储和传输协议，将分布式存储与利用并行性和冗余性的编码缓存相结合，以降低峰值流量速率。

1　研究背景

1.1　系统模型

本文考虑一个 K 个用户和 N 个文件存储在 L 个数据服务器中的网络。当存储数据使用不同的线性组合时，本文还使用了额外的奇偶校验服务器，每个服务器存储相同数量、大小的文件，每个用户都有一个容量为 M 个文件的缓存，并且本文假设所有文件具有相同的长度和流行度。

假设服务器在独立的无错通道上运行，以便两台或两台以上的服务器可以同时传输消息，而不会对相同或不同的用户产生干扰[9]。服务器可以将相同的消息广播给多个用户，而无须额外的带宽成本，但是用户不能共享其高速缓存的内容。同样，每台服务器只能访问正在存储的文件，而不能访问其他服务器上存储的文件。服务器可以从自己的文件中读取多个段，并将它们组合成单个消息，但是存储在不同服务器上的两个文件不能组合成单个消息。但是，假定服务器知道每个用户缓存的内容以及存储在其他服务器中的内容，以便他们能够协调他们的消息[10]。

本文使用子索引来表示文件索引，超级索引来表示段索引，所以 F_i^j 将表示文件 F_i 中的第 j 个段。本文也使用不同的字母来表示来自不同服务器的文件。例如，A_i 表示来自服务器 A 的第 i 个文件，A_i^j 表示来自文件 A_i 的第 j 个段。

1.2　Maddah 方案

Maddah 在文献[6]中提出的编码缓存方案有一台服务器存储所有文件 $\{F_1,F_2,\ldots,F_N\}$，用户通过共享的广播链接连接到该服务器。其目标是设计缓存和交付方案，以减少链路的高峰负载。这个方案将每个文件 F_i 分为 $\binom{k}{t}$ 大小相等的非重叠段 F_i^j，$j=1,2,\ldots,\binom{k}{t}$。其中 $t=\frac{KM}{N}$，并且将每个片段缓存在不同的 t 个用户组中。在交付阶段，服务器向 $t+1$ 个用户的每个子集发送一个消息，总共为 $\binom{k}{t+1}$ 个消息。一组用户 S 请求文件 $F_{i_1},F_{i_2},\ldots,F_{i_{t+1}}$ 将收到消息：

$$m^s=F_{i_1}^{j_1}\oplus F_{i_2}^{j_2}\oplus\cdots\oplus F_{i_{t+1}}^{j_{t+1}} \tag{1}$$

其中：j_k 是除请求 F_{i_k} 的用户之外的所有用户缓存的段的索引。然后，每个用户可以将其缓存中已有的段加密以恢复所需的段。在最坏的情况下，即当所有的用户请求不同的文件时，该方案产生一个(按文件大小标准化)峰值速率为

$$\begin{aligned}R_C(K,t)&=\binom{K}{t+1}/\left(\binom{K}{t}\right)\\&=K(1\text{-}M/N)\frac{1}{1+KM/N}\end{aligned} \tag{2}$$

在一些参数组合下，广播所有未编码的缺失段可能要求比 $R_C(K,t)$ 更低的速率，所以广义峰值速率是：$\min\{R_C(K,t),N-M\}$，假设 N、M 和 K 的关系满足不等式 $R_C(K,t)\le N-M$，本文将忽略那些病态情况。研究表明，这个峰值速率是一些参数组合所能达到的最小值。

1.3　干扰消除

对 Maddah 算法的仔细研究表明，当缓存较小且 $N\le K$ 时，性能较差。因此，文献[7]中提出了一种新的基于干扰消除的编码缓存方案。它不是以简单的形式缓存文件段，而是建议用户缓存多个段的线性组合。所发送的消息被设计为使用最大距离可分码(MDS)来实现[11]。

在存储阶段，这个方案也将每个文件分解为 $\binom{k}{t}$ 个非重叠的大小相同的段。设置 F_i^S 存储，其中 $S\subseteq\{1,2,\cdots,K\}$ 和 $|S|=t$ 表示文件 F_i 中的文件段被 S 中的用户缓存。在放置阶段，用户 k 收集文件段：

$$\{F_i^S\mid i\in\{1,2,\cdots,N\},k\in S\} \tag{3}$$

令 $P=\binom{k-1}{t-1}N$，使用一个长度为 $P_0=2\binom{k-1}{t-1}N-\binom{k-2}{t-2}(N-1)$ 的 MDS 编码 $C(P_0,P)$，且在其缓存中存储 P_0-P 个奇偶校验码。

交付阶段就像所有文件都被请求一样进行。当仅请求一部分文件时，该方案将一些用户的请求替换为“未请求的文件”，并继续进行，如同请求所有文件一样。无论是哪种请求，每个文件 F_i 都传输(未编码或编码)消息。未编码的消息提供了请求 F_i 的用户没有缓存的分段，而组合来自 F_i 的多个分段的编码消息用于消除其缓存段中的干扰。每个用户收集 $\binom{k-2}{t-2}(N-1)$ 个有用的消息，这些消息与存储在其缓存中的 $P-P_0$ 分量一起用来恢复 $C(P_0,P)$ MDS 码中的所有 P 分量。

因此，从服务器发送的消息总数是 $N\binom{k-1}{t}$。在这种干扰消除方案中，可以实现以下标准化 (M,R) 表示：

$$\left(\frac{t[N-1]t+K-N}{K(K-1)},\frac{N(K-t)}{K}\right),t=0,1,\ldots,K \tag{4}$$

1.4　文件条带化

将单服务器编码缓存算法扩展到多服务器系统的最简单方法是将所有服务器上的每个文件分发。这样，每个用户将从每个服务器请求相同数量的信息，平衡负载。这被称为数据条带化[12,13]，在数据中心和固态驱动器中是常见的做法，可以并行写入或读取多个驱动器或存储器块。然后，用户将其缓存的相等部分分配给每个服务器，并且交付按照 L 个独立的单服务器需求来构建。本文现在继续详细描述条带如何降低 Maddah 方案的峰值速率，但同样的想法可以应用于任何其他方案。

N 个文件 $\{F_1,F_2,\ldots,F_N\}$ 被分割成 L 个块存储在不同的服务器中，每个块被分成 $\binom{k}{t}$ 段。这些段由 $F_i^{(j,m)}$ 表示，其中 $i=1,2,\ldots$，N 代表文件编号；$j=1,2,\ldots,\binom{k}{t}$ 代表段号，$m=1,2,\ldots L$ 代表块号。在第 m 个服务器中存储每个文件的第 m 个片段，即每个 i 和 j 的 $F_i^{(j,m)}$。

这个配置和 Maddah 的方案一样。每个段被 t 个用户缓存，$\{F_i^{(j,1)},F_i^{(j,2)},\ldots,F_i^{(j,L)}\}$ 被同一个用户缓存。Maddah 方案可以被分成 L 个分量：

$$F_{i_1}^{(j_1,m)}\oplus F_{i_2}^{(j_2,m)}\oplus\cdots\oplus F_{i_{t+1}}^{(j_{t+1},m)} \tag{5}$$

$m=1,2,\ldots L$ 由不同的服务器发送。 然后将这个问题分解成 L 个独立的单服务器子问题，减少文件大小的 $\frac{F}{L}$ 位。子问题具有与全局问题相同数量的用户、文件和缓存容量。

如果有一个额外的奇偶校验服务器 P 可用，它将为每个文件存储块按位执行 XOR 运算，即对于所有的 i 和 j 均有 $F_i^{(j+1)} \oplus F_i^{(j+2)} \oplus \cdots \oplus F_i^{(j+L)}$。然后，服务器 P 可以接管一些传输，将峰值负载降低到 Maddah 方案的 $\frac{1}{L+1}$。式(5)中，服务器 P 可以传输所有组件的 XOR，以减轻一台数据服务器的传输负担。如果有两个附加的奇偶校验服务器 P 和 Q 可用，则有可能选择 $L+2$ 个服务器中的任何 L 来处理式(5)中的每组消息，从而将 Maddah 方案的峰值降低到 $\frac{1}{L+2}$。

对于干扰消除方案，可以遵循类似的文件分发的处理过程，实现相同的峰值速率缩放比例：无奇偶校验时为 $\frac{1}{L}$，单奇偶校验服务器为 $\frac{1}{L+1}$，两个校验服务器为 $\frac{1}{L+2}$。

1.5 均衡数据放置策略

在分布式文件系统中，当客户端的请求流相对均衡时，如果数据从客户端到存储服务器端的放置也是均衡的，则称整个系统的数据访问相对均衡。可以把数据均衡放置看做是一类映射关系，该映射能够使存储服务器和客户端之间的数据访问保持均衡。

通常在编码存储过程中，以 stripe 作为数据的最小单位来存储，一个大文件被分割为多个 stripe，其中，在每个 stripe 中有 n 个数据块，且前 m 个数据块存储的是原始数据，其余 $n-m$ 个数据块存储的是编码数据，可以把这些数据块当作文件存储对象。则在分布式文件系统中，对象文件的内容由以下部分组成：大文件的 ino 号；每个 stripe 的 $stripe_id$；stripe 内数据块的 $block_id$。比如说，有个 100 GB 的文件，其 ino 号为 2018，在每个 stripe 中有 8 个数据块，原始数据块和编码数据块均为 4 个，设置数据块的容量大小 $block_size$ 为 64MB，则通过计算可知，第 9001MB 数据所处的 $stripe_id$ 是 $9001/(64\times4)=36$，且其所处的 stripe 的 $block_id$ 是 $9001\%(64\times4)/64=0$，此时它所处的对象文件用 $(2018,36,0)$ 三元数组来确定。存储对象的服务器节点 osd_id 由客户端的 $client_id$ 和求得的三元数组 $(ino,stripe_id,block_id)$ 来确定。用 osd_nodes_num 表示文件系统的对象文件节点总数，则每个对象文件的节点 osd_id 表示为：

$$\begin{aligned} osd_id = func0(&func1(ino,stripe_id,block_id) \\ &+client_id)\%osd_nodes_num \end{aligned} \tag{6}$$

其中：映射 $func0:X\to Y$ 以及 $func1:X\to Y$ 定义为 $\forall x\in X:\exists! y\in Y$ 且 $\forall y\in Y:\exists! x\in X$。

通过建立两级映射可以使文件系统在存储大文件时，保证整个系统数据均衡放置，以达到减少磁盘争用的目的。

2 方案 1 大型高速缓存

本章将 Maddah 方案扩展到多服务器系统，结合均衡数据放置策略，将每个文件作为一个单元存储在数据服务器中，如表 1 所示。

表 1 存储在每个服务器的文件

Table 1 Files stored on each server

服务器 A	服务器 B	⋯	服务器 L
A_1	B_1	⋯	L_1
A_2	B_2	⋯	L_2
⋮	⋮		⋮
A_r	B_r	⋯	L_r

Maddah 方案高度依赖于缓存容量 M。与干扰消除相比，Maddah 方案的优点在于文件段以简单形式存储，而不是作为线性组合编码，因此它需要更大的缓存容量来获得编码的缓存增益。因此，当缓存容量很大时，Maddah 的方案是合适的。

本文算法的放置阶段与传统方案中的相同。例如，在用户数 $K=6$，缓存容量为 $M=4$ 和 $N=8$ 个文件的系统中，每个文件被分成 20 个段，每个段由 $t=3$ 个用户存储。表 3 显示了每个用户存储的 10 个文件片段的索引，假设所有文件都是相同的。

考虑到存储系统的峰值速率，本文假设所有用户请求不同的文件，因此每个用户可以用它所请求的文件来表示。用 S 表示为用户集合，m_A^S 表示从服务器 A 发送给 S 中所有用户的消息，$\alpha=\{\alpha_1,\alpha_2,\ldots,\alpha_i\}$ 表示文件索引的向量，$\gamma=\{\gamma_1,\gamma_2,\ldots,\gamma_i\}$ 表示分段索引的向量，$A_\alpha=\{A_{\alpha1},A_{\alpha2},\ldots,A_{\alpha i}\}$ 表示请求或用户，$A_\alpha^\gamma=\{A_{\alpha1}^{\gamma1}\oplus A_{\alpha2}^{\gamma2}\oplus\ldots\oplus A_{\alpha i}^{\gamma i}\}$ 表示消息，其中，A_i^j 代表服务器 A 中第 i 个文件的第 j 个分割，$A_\alpha^\gamma\oplus B_\alpha^\gamma$ 代表消息：

$$A_\alpha^\gamma\oplus B_\alpha^\gamma=\left(A_{\alpha1}^{\gamma1}\oplus B_{\alpha1}^{\gamma1}\right)\oplus\left(A_{\alpha2}^{\gamma2}\oplus B_{\alpha2}^{\gamma2}\right)\oplus\ldots\oplus\left(A_{\alpha i}^{\gamma i}\oplus B_{\alpha i}^{\gamma i}\right)$$

2.1 没有校验服务器

在无冗余的系统中，服务器之间不能协同工作。在交付阶段，每个用户被分配到存储它请求的文件的服务器，然后每个数据服务器发送消息来完成它的请求。它遵循 Maddah 的方案，接收 m 个请求的服务器将需要传输 $\binom{k}{t+1}-\binom{k-m}{t+1}$ 个消息。因此，该服务器的归一化的峰值速率将可以表示为 $\left(\binom{k}{t+1}-\binom{k-m}{t+1}\right)/\binom{k}{t}$，当所有用户请求来自同一个服务器的文件，即 $m=K$ 时，最坏的情况发生，此时峰值传输速率与单个服务器系统相同。

2.2 一个奇偶校验和两个数据服务器

本节介绍一个非常简单的存储系统，它包含两台数据服务器和一台存储上述两个数据按位 XOR 运算的第三台服务器，如表 2 所示。尽管每个服务器只能访问自己的文件，但表 2 中的配置允许通过合并来自任何两个服务器的消息来编写消息。即如果服务器 A(或 B)在另一个服务器之前完成其传输任务，则它可以与奇偶校验服务器一起工作以帮助服务器 B(或 A)。这种协作方案允许同时处理存储在同一服务器上的两个文件请求，平衡负载并将最差情况下的峰值速率降低到小于没有奇偶校验服务器时所达到的峰值速率。

表 2 存储在每个服务器的文件

Table 2 Files stored on each server

服务器 A	服务器 B	服务器 P
A_1	B_1	$A_1\oplus B_1$
A_2	B_2	$A_2\oplus B_2$
⋮	⋮	⋮
A_r	B_r	$A_r\oplus B_r$

还有一个更好的传输方案，将来自三台服务器的消息组合在一起，以便向用户获取更多信息。其基本思想是在数据服务器的每个消息中包含一些未被请求的段以及请求的段。如果附加分段选择得当，则可以将它们与来自奇偶校验服务器的消息组合以获得所需的文件分段。本节提出的算法就是基于这个思想。

就像在 Maddah 的方案中一样，数据服务器把每个消息发送给 $t+1$ 个用户，该消息包含 $t+1$ 个分段。如果用户请求发件人存储的文件，则该消息将包含相应的段；否则消息将包

括它在服务器 P 中的奇偶性的补码，即 A_j，反之亦然。因此，来自服务器 A 或 B 的每个消息的内容由发送者和接收者集合确定，分别用 S_1 或 S_2 表示。在表 3 所示的例子中，从服务器 A 到 $S_1=\{A_1,A_2,A_3,B_4\}$，对应于用户 1 到 4 的消息将是 $m_A^{S_1}=A_1^{11}\oplus A_2^5\oplus A_3^2\oplus A_4^1$。

表 3　文件段到用户缓存的映射

Table 3　File segment mapping to user cache

文件段/用户	1	2	3	4	5	6
1	X	X	X			
2	X	X		X		
3	X	X			X	
4	X	X				X
5	X		X	X		
6	X		X		X	
7	X		X			X
8	X			X	X	
9	X			X		X
10	X				X	X
11		X	X	X		
12		X	X		X	
13		X	X			X
14		X		X	X	
15		X		X		X
16		X			X	X
17			X	X	X	
18			X	X		X
19			X		X	X
20				X	X	X
请求	A_1	A_2	A_3	B_4	B_1	B_2

引理 1　令服务器 A 和 B 的接收者分别是 $S_1=\{A_\alpha,B_\beta,A_*\}$ 和 $S_2=\{A_\alpha,B_\beta,B_*\}$，其中，$\alpha$ 和 β 表示显示的集合，$*$ 代表任意集合，且 $S_1\neq S_2$。相应的信息表示为 $m_A^{S_1}=A_\alpha^*\oplus A_\beta^*\oplus A_*^*$ 和 $m_B^{S_1}=B_\alpha^\eta\oplus B_\beta^*\oplus B_*^*$。选择分段索引，以便每个用户可以取消除了其中一个组件之外的所有组件。这为用户 B_β 和 A_α 分别提供了一些未被请求的段 A_γ 和 B_η。然后服务器 P 可以发送消息 $m_P^{S_1\cap S_2}=(A_\alpha^\eta\oplus B_\alpha^\eta)\oplus(A_\beta^\gamma\oplus B_\beta^\gamma)$ 到 $S_1\cap S_2$，使得 S_1 和 S_2 中的每个用户获得缺失的分段。这三个传输等同于等式(1)中定义的消息 m^{S_1} 和 m^{S_2}。

证明　S_1 和 S_2 中的所有用户至少从存储其请求的文件的服务器获得一个期望的段。那些在 $S_1\cap S_2$ 也收到来自服务器 A 或未请求的段 B。仅证明 $S_1\cap S_2$ 中的用户可以使用这个未请求的段从 $m_P^{S_1\cap S_2}$ 获得它的补码。

不失一般性，考虑用户 $B_{\beta_i}\in S_1\cap S_2$。选择 $m_A^{S_1}$ 中的一组分段索引，以便用户 B_{β_i} 缓存除第 i 个以外的所有分段。同样地，从 $m_B^{S_2}$ 中选择指数 γ 使得 B_{β_i} 可以缓存所有文件。因此，B_{β_i} 可以从 $m_A^{S_1}$ 中获得 $A_{\beta_i}^{\gamma_i}$。将这两者结合得到期望的段 $B_{\beta_i}^{\gamma_i}$。只要 $S_1\neq S_2$，该分段将不同于 B_{β_i} 从 $m_B^{S_2}$ 获得的分段，因为分段索引与用户子集之间存在一对一的关系。

推论 1　假设 $S_1=\{A_*,B_\beta\}$ 并且 $S_2=\{B_*\}$，即它只包含对服务器 B 的请求。服务器 P 向 B 中的所有用户发送 $m_P^{B_\beta}=A_\beta^\gamma\oplus B_\beta^\gamma$，使得 S_1 和 S_2 中的所有用户都获得与 Maddah 方案中相同的分段。

证明　当 α 为空(β 可以是空的或非空的)时，这是引理 1 的一个特例。

定理 1　如果用户子集 S_1 和 S_2 满足引理 1 中的条件，本文称 (S_1,S_2) 是一个有效的对。

本文的目标是设计一个方案，它能够最小化任何服务器发送的最大消息数量。如果两个用户子集形成一个有效的对，则可以用每个服务器的单个传输来替换 Maddah 方案中相应的消息。因此，本文希望尽可能多地作出有效的配对。

引理 2　表 2 中服务器系统的峰值速率为 $\left(\frac{1}{2}+\frac{1}{6}\Delta\right)R_C(K,t)$，其中 Δ 表示未配对消息的比率，并且 $t=KM/N$。

证明　对于每个有效的配对，本文可以使用来自每个服务器的单个传输来提供与在 Maddah 的单服务器方案中的两个传输相同的信息，速率为 $\frac{1}{2}(1-\Delta)R_C(K,t)$。未配对的消息按照上文的描述进行传输，即从三台服务器中的任何两台服务器合并消息。假设这三个服务器之间的负载是平衡的，则速率变为是 $\frac{2}{3}R_C(K,t)$。

下面的引理描述了在对称请求的情况下(两个服务器接收到相同数量的请求)不成对的用户子集 Δ 的比率。

引理 3　如果请求是对称的，则当 t 是偶数时 $\Delta=0$，当 t 是奇数时 $\Delta\leq 1$。也就是说，在对称请求的情况下，可以达到以下的峰值速率：

$$R_T(K,t)=\begin{cases}\frac{1}{2}R_C(K,t),\text{偶数}\\ \left(\frac{1}{2}+\frac{1}{6}\Delta\right)R_C(K,t),\text{奇数}\end{cases}\tag{7}$$

其中：$R_C(K,t)$ 在式(2)中已经定义。

在表 3 中，每个分段被 $t=\frac{KM}{N}=3$ 个用户缓存，配对算法的标准化峰值率为 2/5，明显低于 Maddah 单服务器方案的 3/4。

2.3　一个奇偶校验和 L 个数据服务器

前面讨论了两个数据服务器和一个奇偶校验服务器的情况，相同的算法可以扩展到具有两个以上数据服务器的系统。如果有 L 个数据服务器和一个奇偶校验服务器，则可以通过合并来自 L 个服务器的消息来建立消息。假设 $S_1=\{A_\alpha,B_\beta,A_*,Y\}$，$S_2=\{A_\alpha,B_\beta,B_*,Y'\}$ 为两个用户子集，其中 Y 和 Y' 为服务器 C 通过 L 和 $*$ 表示的任意索引集合，并且 S_1 和 S_2 可以配对，这样服务器 A，B 和 P 需要一次传输来提供与 Maddah 单服务器方案中的消息 m^{S_1} 和 m^{S_2} 相同的信息。其他数据服务器 C 到 L 需要最多两次传输，如图 1 中的成对传输所示，A、B、C 和 D 是数据服务器，P 代表奇偶校验服务器。X 表示由相应的服务器发送消息。

整个传输过程如下：

a）服务器 C 发送两个消息到 S_1 和 S_2。例如，服务器 C 想要发送 $m_C^{S_1}$ 和 $m_C^{S_2}$，向相应的资源用户发送请求信息。

b）服务器 A 发送 $m_A^{S_1}$，向请求 $\{A_*,A_\alpha\}$ 的用户提供所需的段，并向请求 B 的用户提供相应的不需要的 A 段。

c）服务器 B 发送 $m_B^{S_2}$，向请求 $\{B_\beta, B_*\}$ 的用户提供所需的段，并向请求 A_α 的用户提供相应的不需要的 B 段。

d）服务器 P 向请求的用户发送 $m_P^{\{A_\alpha,B_\beta\}}$。使用之前接收到的不需要的段，$\{A_\alpha,B_\beta\}$ 中的用户可以求解所需的 A 段和 B 段。

对所请求和接收的段进行简单的比较表明，这些传输在 Maddah 的单服务器方案中传递与消息 m^{S_1} 和 m^{S_2} 相同的信息。

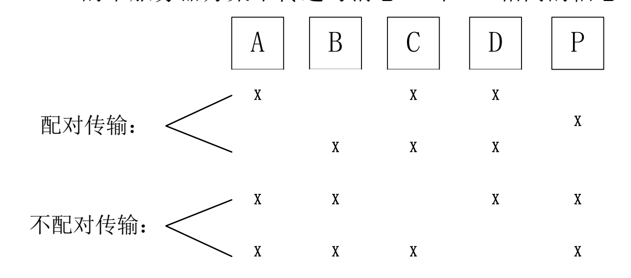


图 1　配对 4 个数据服务器和一个奇偶校验服务器系统

Fig. 1　Matches four data servers with a parity server system

定理 2　具有 $L\geq 3$ 个数据服务器和一个奇偶校验服务器的系统可以达到以下标准化峰值速率：

$$R_P(K,t)=\frac{L-1}{L}R_C(K,t) \tag{8}$$

证明　在 Maddah 方案中，可以从服务器 A、B 和 P 中发送 $\frac{2}{L}\binom{K}{t+1}$ 个消息，这些消息可以按照如下方式发送完成：按照它们所具有的段的数量对消息进行分组，在每个组内进行消息配对，每组中至少有 $\frac{2}{3}\geq\frac{2}{L}$ 个消息可以配对。没有 A 或 B 分段的消息不需要从服务器 A、B 或 P 进行任何传输。剩余的 $\frac{L-2}{L}\binom{K}{t+1}$ 个消息可以通过服务器 C 传输。

2.4　两个奇偶校验和 L 个数据服务器

本节把算法扩展到具有 L 数据和两个线性奇偶校验服务器的系统。奇偶校验服务器通过行来存储文件的不同线性组合，如表 4 所示。假定服务器形成 MDS 码。本文交付策略表明，峰值速度可以降低到 Maddah 单服务器方案的一半。

表 4　在 RAID-6 校验服务器中存储的文件

Table 4　Files stored in RAID-6 Check Server

服务器 P	服务器 Q
$A_1+B_1+\ldots+L_1$	$A_1+k_BB_1+\ldots+k_LL_1$
$A_2+B_2+\ldots+L_2$	$A_2+k_BB_2+\ldots+k_LL_2$
$\vdots$	$\vdots$
$A_r+B_r+\ldots+L_r$	$A_r+k_BB_r+\ldots+k_LL_r$

引理 4　设 $S_1=\{A_*,Y\}$，$S_1=\{B_*,Y\}$，其中 Y 代表来自任何服务器的一组公共请求。S_1 和 S_2 可以配对，以便每个服务器的单个传输填充相同的请求作为式(1)中的消息 m^{S_1} 和 m^{S_2}。

证明　传输方案与第 2.2 小节中的算法有相同的配对思路。传输过程如下：

a）服务器 A 发送 $m_A^{S_1}$，向请求其文件的用户提供期望的段，并向其他用户提供相应的非期望的 A 段。

b）服务器 B 发送 $m_B^{S_2}$，向请求其文件的用户和对应的不需要的 B 段提供所需段。

c）服务器 $C,D,\ldots,L$ 每个用户向 $S_1\cap S_2=\{Y\}$ 发送单个消息，每个用户具有以下内容：

(a)从服务器 B 请求文件的用户从服务器 A 收到一些不需要的段。服务器 $C,D,\ldots,L$ 向它们发送匹配的数据，以便期望的段可以稍后使用服务器 P 中的奇偶校验来解码。

(b)Y 中的剩余用户将在可能的情况下获得与 S_1 相应的期望段，否则将得到与 S_2 相对应的不需要的段。

换句话说，每个服务器 $C,D,\ldots,L$ 将发送对应于 S_1 的片段给请求其文件的用户或来自服务器 B 的片段。此时，所有用户都满足了与 S_1 相关的请求，而从服务器 B 请求文件的用户满足了与 S_2 相关的请求。 每个用户也收到了 $L-2$ 个不需要的“匹配”的片段。

d）奇偶校验服务器 P 和 Q 使用每个用户的段的组合向 $S_1\cap S_2=\{Y\}$ 发送消息。那些从服务器 B 请求的文件将得到两个对应于 S_1 的段的组合，其余的将得到对应于 S_2 的段的组合。 由于每个用户现在都有 $L-2$ 个独立的段和所有 L 段的两个独立的线性组合，它可以隔离所请求的段。

对所请求和接收的段进行比较可知，这种传输方式具有和 Maddah 单服务器方案相同的消息 m^{S_1} 和 m^{S_2}。

定理 3　对于 L 个数据服务器和两个奇偶校验服务器系统，可以实现以下归一化峰值速率：

$$R_Q(K,t)=\left(\frac{1}{2}+\frac{L-2}{2L+4}\right)R_C(K,t) \tag{9}$$

其中：$\Delta\leq\frac{1}{3}$ 是配对损失，R_C 是等式(2)中单服务器 Maddah 方案的比率。

证明　按照它们从服务器 A 或 B 所具有的段的数量对消息进行分组。在每个组内将消息配对。对于不包含来自 A 或 B 的段的消息，本文重复与其他两个服务器相同的过程，得到相同的结果：最多只有 1/3 个未配对。

如上文所示，每对消息可以使用来自每个服务器的单个传输来传递，因此配对消息速率为 $\frac{1}{2}(1-\Delta)R_C(K,t)$，其中 Δ 表示未配对消息的比例。在所有服务器之间平衡负载，它们的速率为 $\frac{L}{L+2}\Delta R_C(K,t)$。

3　方案 2　小型缓存

将 1.3 节中的干扰消除方案扩展到多服务器系统，结合均衡数据放置策略，提出缓存的线性组合。干扰消除方案专门设计用于在高速缓存大小较小时降低峰值速率。表 1 表明，传输速率随着服务器数量 $\frac{1}{L}$ 的减少而减少。对奇偶校验服务器的情况进行类似的分析，这可以解释为用户缓存的扩展。

定理 4　在具有 L 个数据服务器和并行信道的系统中，干扰消除方案的峰值速率可以降低到单服务器系统中的 $\frac{1}{L}$，即可以实现以下 (M,R) 对：

$$\left(\frac{t[N-1]t+K-N}{K(K-1)},\frac{N(K-t)}{K}\right),t=0,1,\ldots,K \tag{10}$$

无论每个文件是跨服务器(分条)还是作为单个块存储在一台服务器中，都是如此。

证明　与单个服务器系统相比，L 个服务器上的文件条带化将干扰消除方案的峰值速率降低了 $\frac{1}{L}$。

与 Maddah 的方案相比，干扰消除方案从每个文件发送相同数量的段，而不管用户的请求。此外，每个消息由一个文件组成的段组成。因此，即使不同的文件存储在不同的服务器中，也可以传输相同的消息。每个服务器将需要传输 $\frac{1}{L}$

小部分的消息，因为它将存储相同比例的文件。然后峰值负荷可以减少到式(4)中的 $\frac{1}{L}$ 。

如果有奇偶校验服务器，本文可以通过将它们作为用户缓存的扩展来进一步降低传输速率。第 1.3 节解释了在干扰消除算法中，每个用户缓存由系统 MDS 码 $C(P_0,P)$ 编码一组分段产生的奇偶校验符号。可以按照这样的方式选择代码，即可以将这些奇偶校验符号中的一些作为存储在服务器 P 和 Q 中的信息的组合来找到。

例如，奇偶校验服务器 P 存储文件的水平和，所以它可以传送以下格式的消息：$\sum_{i=1}^{N/L}\sum_{j=1}^{\binom{K-1}{t}}\lambda_{ij}(A_i^{S_j}+B_i^{S_j}+\ldots+L_i^{S_j})$ 。对于任何用户集合 s_j 具有任意系数 λ_{ij} 。这对应于方程(3)中所有分段的线性组合。类似地，奇偶校验服务器 Q 可以传输一些线段的其他线性组合，也可以作为 MDS 码的组成部分。这有效地增加了 M' 个文件单元的高速缓冲存储器的大小，对应于奇偶校验服务器在传送阶段能够发送给每个用户的信息量。

定理 5 如果有 η 个奇偶校验服务器且 $K\ge N$ ，则对于 $t=0,1,\ldots,K$ ，可以得到以下 (M,R) 对：

$$\left(\frac{t[N-1]t+K-N}{K(K-1)}-\eta\frac{N(K-t)}{LK^2},\frac{N(K-t)}{LK}\right) \tag{11}$$

证明 由奇偶校验服务器发送的信息受到数据服务器的峰值速率的限制，即根据等式(10)的 $\frac{N(K-t)}{LK}$ 。假设最坏的情况，从奇偶校验服务器的每个传输将有益于单个用户。因此，每个奇偶校验服务器可以通过 $M'=\frac{N(K-t)}{LK^2}$ 有效地增加每个用户的缓存。

这种内存共享策略在缓存容量很小时提供了显著的改进。图 2 显示了存储在 $L=4$ 个数据服务器中的 $K=15$ 个用户和 $N=12$ 个文件的性能。当缓存容量 M 较小时，带有两个奇偶校验服务器的系统的峰值速率 R 远远低于没有奇偶校验服务器的情况。随着缓存的增长，使用奇偶校验服务器的系统的优势变得越来越小。

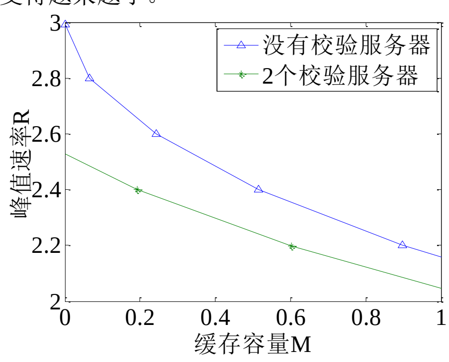


图 2 没有奇偶校验服务器和使用两个奇偶校验服务器的性能

Fig. 2 Shows no parity server and performance using two parity servers

干扰消除方案是针对单服务器系统中文件数少于用户($N\le K$)的情况而设计的。但是，由于在多服务器系统中，峰值负载减少了 $\frac{1}{L}$ ，所以当 $L>N$ 时，如果 $N>K$ ，干扰消除方案也可能具有良好的性能。为了应用该算法，本文可以只添加 $N-K$ 个具有任意请求的虚拟用户。那么，本文有以下结论：如果有 η 个奇偶校验服务器且 $K\le N$ ，则可以实现以下 (M,R) 对：

$$\left(\frac{t^2}{N}-\eta\frac{(N-t)}{LN},\frac{(N-t)}{L}\right),t=0,1,\ldots,N \tag{12}$$

4 仿真实验与分析

通过基于 Linux 的 NS2 仿真软件对提出的两种方案进行仿真实验，共设置了两组实验，两组实验均在一个奇偶校验服务器系统和两个奇偶校验服务器系统中进行，并将提出的方案 1、方案 2 与 Maddah 方案条带化[6]、干扰消除条带化[7]进行比较，接下来简单介绍实验参数设置。

4.1 实验参数设置

两组实验分别设置不同的 N 和 K 值，第 1 组设置为 $N=20$ ， $K=15$ ，第 2 组设置为 $N=20$ ， $K=60$ 。

对于一个系统， $N=20$ 个文件存储在 $L=4$ 个数据服务器中，每个服务器有 5 个文件，用户组 t 设置为 5，方案 1 和 2 的参数计算式分别如表 5、6 所示，可根据不同的 N 、 K 、 t 值值计算得到相应的服务器参数值。

表 5 方案 1 的归一化峰值速率

Table 5 Normalized peak rate of scheme 1

服务器系统	归一化峰值速率
单个服务器	$R_C(K,t)=\binom{K}{t+1}/\binom{K}{t}$
L 个数据，一个奇偶校验	$\frac{L-1}{L}R_C(K,t)$
L 个数据，两个奇偶校验	$\left(\frac{1}{2}+\frac{l-2}{2L+4}\Delta\right)R_C(K,t)\left(\Delta\le\frac{1}{3}\right)$

表 6 方案 2 的归一化 (M,R) 对

Table 6 Normalized (M,R) pairs of scheme 2

服务器系统	归一化 (M,R) 对
单个服务器	$\left(\frac{t[N-1]t+K-N}{K(K-1)},\frac{N(K-t)}{K}\right)$
L 个数据， η 个奇偶校验	$\left(\frac{t[N-1]t+K-N}{K(K-1)}-\eta\frac{N(K-t)}{LK^2},\frac{N(K-t)}{LK}\right)$
L 个数据， η 个奇偶校验	$\left(\frac{t^2}{N}-\eta\frac{(N-t)}{LN},\frac{(N-t)}{L}\right)$

4.2 实验结果和分析

图3和4分别显示了一个和两个奇偶校验服务器的情况。假设有 K=15 个用户， N=20 个文件。由图可知，条带化提供了更低的峰值速率。另外，由于 $N>K$ ，在使用条带化时，干扰消除方案总是比 Maddah 方案具有更差的性能。在没有条带化的情况下，当缓存容量较小时，方案 2 的峰值速率比方案 1 更低。

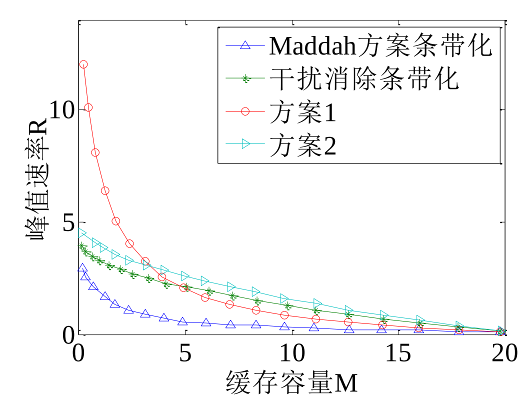


图 3 N=20， K=15 时，一个奇偶校验服务器系统中四种方案的性能

Fig. 3 N=20 , K=15 . Performance of four schemes in a parity-check server system

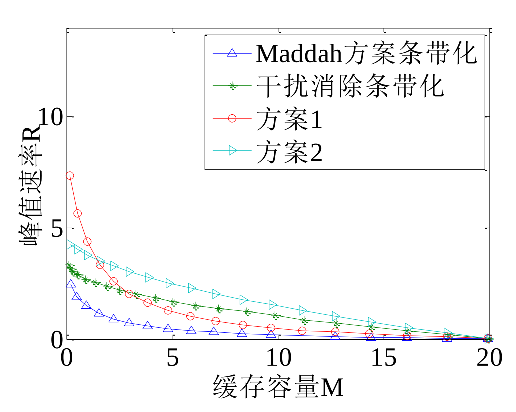


图 4　N=20，K=15 时，在两个奇偶校验服务器系统中四种方案的性能

Fig. 4　N=20 , K=15 . Performance of four schemes in two parity-check server systems

图 5 和 6 分别比较了一个或两个奇偶校验情况下用户数为 $K=60$ 时方案 1 和方案 2 的性能。同样的，条带化提供了更低的峰值速率，且当缓存容量很小时，条带化干扰消除比条带化 Maddah 方案有更好的性能。对于方案 1 和方案 2，当缓存容量较小时，方案 2 提供较低的峰值速率，而当缓存容量增加时，方案 1 具有较好的性能。而且，曲线在 M 的交叉点比在图 3 和 4 中的点更大，这意味着当用户数量多于文件数量时，更倾向于使用方案 2 来进行缓存。

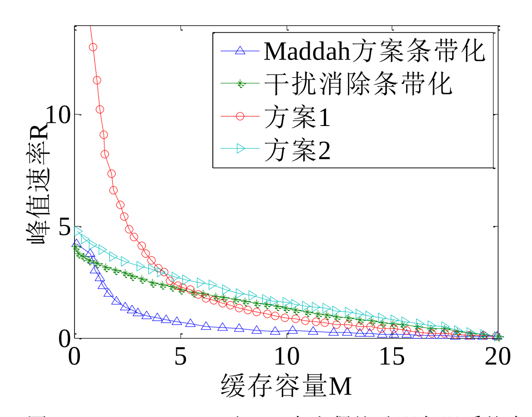


图 5　N=20，$K=60$ 时，一个奇偶校验服务器系统中四种方案的性能

Fig. 5　N=20，$K=60$. Performance of four schemes in a parity server system

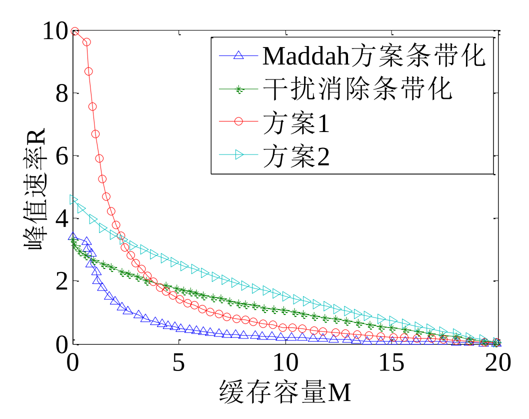


图 6　N=20，$K=60$ 时，两个奇偶校验服务器系统中四种方案的性能

Fig. 6　Performance of four schemes in two parity-check server systems

5　结束语

本文提出一种用于降低具有分布式存储和不同冗余级别的多服务器系统中的峰值数据速率的编码缓存方案，通过在多个服务器上分割每个文件，峰值速率可以与服务器数量成比例地减少。当每个文件作为单个块存储在一个服务器中时，算法根据高速缓存存储器的大小提供了不同的高速缓存和交付方案。本文提出的编码缓存方案能够以创造性的方式利用这种冗余来降低可达到的流量峰值速率。下一步的工作是设计具有不同文件保护级别的擦除码[14,15]，以便将本文方案推广到具有不同普及性的文件系统中。

参考文献：

[1] 卞建超，查雅行，罗守山，等. 一种基于磁盘内和磁盘间冗余的混合编码方案 [J]. 计算机研究与发展，2016, 53(9): 1906-1917. (Bian Jianchao, Chaya Hang, Luo Shoushan, et al. A hybrid coding scheme based on intra-disk and inter-disk redundancy [J]. Computer Research and Development, 2016, 53(9): 1906-1917.)

[2] 罗平，张彤. 基于闪存的星载存储数据管理研究 [J]. 计算机应用研究 ,2018,35(2):479-482. (Luo Ping, Zhang Tong. Research on spaceborne storage data management based on flash memory [J]. Applition Research of Computers, 2018, 35(2): 479-482.)

[3] Zhang Guangyan, Wang Jigang, Li Keqing, *et al.* Redistribute data to regain load balance during RAID-4 scaling [J]. IEEE Trans on Parallel & Distributed Systems, 2014, 26(1): 219-229.

[4] Wang Yan, Yin Xunrui, Wang Xin. MDR Codes: A new class of RAID-6 codes with optimal rebuilding and encoding [J]. IEEE Journal on Selected Areas in Communications, 2014, 32(5): 1008-1018.

[5] Balasubramanian B, Lan T, Chiang M. SAP: similarity-aware partitioning for efficient cloud storage [C]// Proc of INFOCOM.Piscataway,NJ: IEEE Press, 2014: 592-600.

[6] Maddah-Ali M A, Niesen U. Fundamental limits of caching [J]. IEEE Trans on Information Theory, 2012, 60(5): 2856-2867.

[7] Ghasemi H, Ramamoorthy A. Improved lower bounds for coded caching [J]. IEEE Trans on Information Theory, 2017, 63(7): 4388-4413.

[8] Tian C, Chen J. Caching and delivery via interference elimination [C]// IEEE International Symposium on Information Theory. Piscataway,NJ: IEEE Press, 2016.

[9] 邓燕，张新有，邢焕来. 一种基于传输容量控制的 DTN 动态分段编码路由算法 [J]. 计算机应用研究，2017, 34(9): 2753-2757. (Deng Yan, Zhang Xinyou and Xing Huanlai. A DTN dynamic segment coding routing algorithm based on transmission capacity control [J]. Application Research of Computers, 2017, 34(9): 2753-2757.)

[10] 朱红，李立，黄普明. 星载海量遥感数据的低缓存高速传输[J]. 电子学报，2013, 41(10): 2016-2020. (Zhu Hong, Li Li, Huang Puming. Low-cache and high-speed transmission of massive satellite-borne remote sensing data [J]. Journal of Electronics, 2013, 41(10): 2016-2020.)

[11] Suh C, Ramchandran K. Exact-repair MDS code construction using interference alignment [J]. IEEE Trans on Information Theory, 2011, 57(3): 1425-1442.

[12] 董秋香，关志，陈钟. 加密数据上的计算密码学技术研究综述 [J]. 计算机应用研究，2016, 33(9): 2561-2572. (Dong Qiuxiang, Guan Zhi, Chen Zhong. Review of computational cryptography on encrypted data [J]. Application Research of Computers, 2016, 33(9): 2561-2572.)

[13] Shen Xiaohui, Choudhary A. A high-performance distributed parallel file system for data-intensive computations [J]. Journal of Parallel & Distributed Computing, 2004, 64(10): 1157-1167.

[14] Yu Quan, Sung C W, Chan T H. Irregular fractional repetition code optimization for heterogeneous cloud storage[J]. IEEE Journal on Selected Areas in Communications, 2014, 32(5): 1048-1060.

[15] Yin Chao, Wang Jianzong, Lv Haitao, *et al.* BDCode: an erasure code algorithm for big data storage systems [J]. Journal of University of Science & Technology of China, 2016, 46(3): 188-199.